Secure VPNs for Trusted Computing Environments

Steffen Schulz and Ahmad-Reza Sadeghi

Horst-Görtz Institute and Chair for System Security, Ruhr-University Bochum
{steffen.schulz,ahmad.sadeghi}@trust.rub.de

Abstract. Virtual Private Networks are a popular mechanism for building complex network infrastructures. Such infrastructures are usually accompanied by strict administrative restrictions on all VPN endpoints to protect the perimeter of the VPN. However, enforcement of such restrictions becomes difficult if these endpoints are personal computers used for remote VPN access. Commonly employed measures like anti-virus or software agents fail to defend against unanticipated attacks. The Trusted Computing Group invested significant work into platforms that are capable of secure integrity reporting. However, trusted boot and remote attestation also require a redesign of critical software components to achieve their full potential.
In this work, we design and implement a VPN architecture for trusted platforms. We solve the conflict between security and flexibility by implementing a self-contained VPN service that resides in an isolated area, outside the operating system environment visible to the user. We develop a hardened version of the IPsec architecture and protocols by addressing known security issues and reducing the overall complexity of IPsec and IKEv2. The resulting prototype provides access control and secure channels for arbitrary local compartments and is also compatible with typical IPsec configurations. We expect our focus on security and reduced complexity to result in much more stable and thus also more trustworthy software.

1 Introduction

VPNs are a simple and cost effective way to manage and control complex networks. With increasing user mobility however, the VPN perimeter also becomes increasingly complex. Mobile systems are expected to serve as secure VPN gateways, workstations and personal devices at the same time. As a result, it becomes increasingly difficult to assure the security of such systems. Allowing them to connect to a VPN potentially undermines perimeter security and may expose the network to outside attacks. Vendors try to solve this conflict with proprietary security software and software agents, but such solutions increase complexity of the software stack while decreasing interoperability with other software solutions.

With trusted computing, the state of a system can be measured at boot time. A hardware anchor is used to vouch for the correctness of measurement reports, so that the integrity of a system can be verified by remote parties before granting any kind of access. The measurement itself however is not sufficient to trust the system. The integrity

Full version. The original publication is available at www.springerlink.com of a software configuration is only useful if the software itself can be trusted to fulfill the security requirements. This implies a resistance to attacks and misconfiguration that current commodity systems do not achieve. We follow architecture proposed in [1] to separate volatile userspace environments from components that are critical to system security and to allow strong isolation of userspaces for the different roles assumed by the user.

An architecture similar to ours is described in in [2]. However, their work does not consider integration with trusted computing technologies. and proved unsuitable for our work to build up on.

1.1 Contribution

This work adapts the IPsec security architecture for a robust and reliable VPN service for trusted platforms. In our design, critical functionality is externalized into isolated, self-contained security services in a trusted hypervisor environment. We use a central security policy from trusted storage to establish secure channels between isolated userspace environments and to connect them to other IPsec networks. As a result, our architecture enables coexistence of arbitrary userspace environments with restricted workspaces while reliably enforcing the platform owner's network access policy.

To harden our VPN security service, we investigate a simplified IPsec architecture supporting only tunnel mode with ESP[1] protection [3] and IKEv2[2] key negotiation [4]. By removing unnecessary functionality and features that allow insecure IPsec operation modes, we resolve known security issues in IPsec and significantly reduce the complexity of architecture and implementation.

We have implemented a prototype based on the L4 microkernel to verify the feasibility of our solution, to evaluate its interoperability with commodity IPsec implementations, and to measure the complexity in terms of code size. Finally, we discuss compatibility and security of our architecture and suggest feature improvements.

1.2 Applications

Our architecture has significant security-benefits for users that assume different roles during their work. The typical example for home users is online banking, where the sVPN architecture allows to setup a fully isolated userspace environment with strong administrative restrictions to secure banking sessions. Similar use-cases can be found in corporate environments, where access to critical network resources is often restricted to machines with specific software configurations. In such setups, the sVPN service resolves the conflict between flexibility and security by moving all critical functionality out of the userspace environment. Once we finish our integration with trusted storage and remote attestation, our architecture also allows to verify the state of a peer's security subsystem, enforce arbitrary security requirements on connected userspaces. Additionally, our design also delivers a very simple and usable way to deploy secure and IPsec-compatible VPNs. Since our design also focuses on minimal complexity for the

[1] Encapsulated Security Payload
[2] Internet Key Exchange version 2.

critical components, environments with high security demands may also benefit from the possibility of formal code-reviews of the critical components of their VPN. Finally, our implementation also provides virtualized IPsec gateways that can be used to consolidate hardware resources in more complex VPN setups.

1.3 Outline

We present the general idea of our architecture in section 2, followed by a requirements analysis in section 3. In section 4, we investigate related VPN designs and security architectures for microkernel environments. Section 5 reviews the security of the IPsec architecture we leverage on. Based on these results, we design a secure compartmentalized VPN service in in section 6. Section 7 presents our prototype implementation and compares its code-complexity with standard implementations. In sections 8 and 9, we discusses how our modifications influence compatibility and security. We conclude with summary of results and suggestions for future work.

2 High-Level Architecture

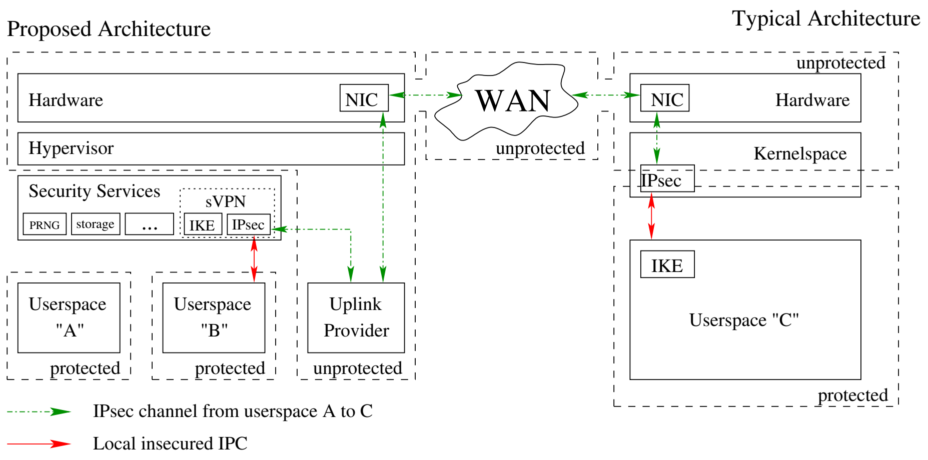

Fig. 1. Overview of the sVPN architecture interfacing with a commodity IPsec implementation through WAN/Internet. The sVPN security service processes all data passing through a local channel between "B" and the "Uplink Provider", according its IPsec policy. By applying IPsec protection to some of the data streams, local channels are logically extended into secure remote channels that reach through the unprotected area to a peer IPsec gateway.

Figure 1 shows the VPN architecture of two platforms connected through a wide area network (WAN). The right system shows a commodity IPsec implementation, while the left system shows the design proposed in this work.

In commodity systems, IPsec packet processing is typically implemented as part of the network stack in the kernel. An *IPsec boundary* enforces the IPsec security policy (`BYPASS`, `DISCARD`, `PROTECT`) on all traffic that passes through it, dividing the system into a "protected" and "unprotected" area. Userspace environments (compartments) in the "protected" area use the VPN service by specifying the `PROTECT`-target on specific traffic flows. The IPsec boundary then provides a secure channel through the "unprotected" area. The channel ends at a peer IPsec boundary, which decapsulates the data and forwards it to its local "protected" area. In such systems, key negotiation is typically handled by the "protected" endpoints of the channel. With only a single IPsec boundary, such systems also support only a single "protected" area that must be shared between all userspace applications.

The left-hand system in contrast provides multiple isolated userspace environments by design. Virtualization of legacy operating systems and basic operating system functionality is provided by a trusted hypervisor, which is accompanied by an environment of system and security services. Since the different userspace environments have potentially conflicting security requirements, they build separate "protected" areas in IPsec that require a dedicated logical IPsec boundary for policy enforcement. Similarly, the key negotiation component is not part of any particular userspace environment anymore but must be implemented as a neutral security service of the system. Both components are thus implemented as self-contained security services in the trusted hypervisor environment. They are configured by the platform owner and do not rely on any untrusted components to maintain their security. Once packets are processed, the "unprotected" area has the non-critical task of delivering the data to the destination IPsec gateway. Figure 1 thus only shows a single common *uplink provider* for post-processing and uplink management.

The proposed compartmentalized architecture is called *sVPN architecture* throughout this paper, our prototype implementation of it is called *sVPN service* or just *sVPN*. In includes the two mentioned security services in the hypervisor environment and some untrusted applications for pre- and post-processing.

3 Requirement Analysis

3.1 Functional Requirements

IPsec Virtualization In contrast to typical IPsec implementations, the sVPN service has to be able to provide its VPN service for multiple "protected" areas with potentially conflicting security requirements.

`/R1/` sVPN must establish bidirectional channels between local compartments and manage a separate IPsec security policy database (SPD) and security association database (SAD) for each channel.

Usability To let users to benefit from the enhanced security of our design, it is necessary that the user interface provides high usability and prevents typical configuration errors.

`/R2/` sVPN must provide a usable VPN configuration interface that is able to define and deploy secure VPNs and prevents accidental use of insecure cryptographic primitives, operation modes or authentication schemes.

`/R3/` sVPN must depend on as few components as possible to implement its security. These dependencies must be specified and easy to understand.

Compatibility Interoperability with other IPsec-based VPN solutions is strongly desired, as long as it does not conflict with the security of the VPN service.

`/R4/` sVPN must be compatible with IPsec in typical VPN configuration.

`/R5/` sVPN must provide basic support for the canonical key exchange protocol for IPsec VPNs (i.e., IKEv2).

Security Subsystem We optimize our VPN service for low internal complexity to facilitate formal security analysis and to reduce the possibility of security flaws in the implementation.

`/R6/` The sVPN architecture must isolate components that must be trusted to meet the security requirements.

`/R7/` All critical subcomponents and the services they depend on must be of manageable internal complexity with simple communication interfaces.

3.2 Security Requirements

Adversary Model To simulate the susceptibility of complex applications to local and remote attacks, we assume an adversary to have full access to all userspaces a legal user of the platform has access to. In contrast to legal users however, adversaries are assumed to have no physical access to the platform. An adversary is considered successful if he manages to extract secret key material from the sVPN security service or if he is able to violate the sVPN security policy, for example by creating additional channels between local compartments. In addition, when attacking only from "unprotected" compartments or networks, i.e. without implicit knowledge of transmitted data, he is also considered successful if he manages to extract data that is labeled with `PROTECT` in the sVPN security policy.

The attack model for the sVPN security service is stronger than that of traditional IPsec-based VPN solutions. The main difference is that local compartments which interface with sVPN are assumed to be compromised, thus providing an attacker with reliable access to data transfers and statistics about resource usage to launch side-channel attacks (e.g. timing attacks, traffic pattern analysis). We assume however that isolation provided by the hypervisor also protects against side-channel attacks based on shared resources[5, 6]. Therefore, we only considers side-channel attacks in form of traffic pattern analysis.

Further, we assume that the IPsec architecture has no security flaws aside from those mentioned in section 5.1, i.e. that the core IKEv2 protocol, the ESP protocol and the IPsec architecture itself are sound. Based on these assumptions, we identify the following requirements for the sVPN architecture to remain secure in the described attack model.

Virtualization Security sVPN has to provide a logically isolated IPsec boundary for every compartment that uses the service, thus isolating every connected userspace environment into a separate "protected" area.

/S1/ sVPN must be able to identify endpoints of a local channel and must enforce the corresponding IPsec policy for that channel.
/S2/ Communication channels between local compartments that are not directed through sVPN must be restricted by the hypervisor such that they can not be used to circumvent the security enforced by sVPN.
/S3/ Although acting as a common endpoint for multiple local channels, sVPN must not break isolation between compartments that are not allowed to establish a channel between each other.

IPsec Security sVPN must provide a secure VPN service through secure authentication, key management and secure channels.

/S4/ The sVPN architecture must provide a secure channel for deploying IPsec policy and authentication secrets to the sVPN service.
/S5/ sVPN must never disclose any key data to untrusted components. Accessibility of key material must be minimized to the necessary sub-components of sVPN.
/S6/ sVPN must be able to directly authenticate the user of a "protected" area when required so by IPsec policy.
/S7/ The IPsec compatible remote channels provided by sVPN must be secure.
/S8/ sVPN must be able to enforce restrictions on the configuration of compartments that request a particular VPN access.

Since non-critical functionality of sVPN, including receiving and sending of data, is externalized to the connecting "protected" and "unprotected" compartments which are expected to be compromised, it is not possible to protect against DoS in the adversary model described above. If a DoS resistance similar to commodity IPsec implementations is desired, it can thus be implemented in the traffic pre-processing in untrusted compartments.

4 Related Work

Hypervisor-based operating systems environments with userspaces on multiple virtual machines have recently been reconsidered as a base for increased security. The idea to externalize security subsystems into a hypervisor environment resulted in several new architectures like sHype, Terra, EROS, Nizza and Perseus [7–10, 1, 11].

A conceptionally similar setup can be found in many microkernel operating systems, but their current IPsec implementations do not exploit the features of their environment for additional security. They exist only in form of adapted versions from monolithic systems or university classes [3].

[3] See for example `www.cis.syr.edu/~wedu/seed/Labs/IPSec/`, where simple IPsec processing is regularly implemented on Minix 3.

With µSINA, the authors of [2] present a comparable redesign of IPsec for Nizza. In their architecture, a "network hub" is added to the operating system environment of a Fiasco/L4 microkernel. It processes traffic between two paravirtualized Linux compartments that run on top of the microkernel (L4Linux compartments), enforcing IPsec policies and traffic protection. µSINA aims for enhanced reliability and security by minimizing the attack surface of critical components. An implementation of the IPsec packet processing component is provided with the *Viaduct* security service.

µSINA was later supplemented by a key negotiation component [12]. To manage the high complexity of the IKEv1 protocol, this component was implemented as a port of the *isakmpd* server from the OpenBSD project. Two adapter components for translation to Unix sockets and the servers native management interface PF_KEYv2 [13] where added to simplify the port.

However, the Viaduct source code is difficult to read, poorly documented and contains considerable amount of non-critical code, for example to route packets between compartments. We were also unable to test it since the development was discontinued and does not work with the current DROPS environment. From the descriptions of the IKEv1 implementation in [12], it is also obvious that µSINA suffers from the high complexity of the IKEv1 standard.

In contrast to µSINA, we do not aim for a generic IPsec implementation. As detailed in the following sections, we propose a more abstract VPN service instead that implements only a reduced set of the IPsec functionalities and exploits the potential of secure virtualization and trusted computing architecture. The resulting design solves many common security issues of VPN endpoints and maximizes the leverage on policy enforcement available through remote attestation.

5 IPsec Security

The IPsec security standard was first specified in 1995 [14]. It is a collection of Internet standards that provide access control, integrity protection and authentication, confidentiality and partial protection against packet replay.

The Internet Engineering Task Force (IETF) published the latest version of the architecture in [15], featuring mainly a simplified design and description. The associated Internet Key Exchange (IKE) protocol was subject to a major redesign and published as version 2 (IKEv2) in [4]. IPsec uses two protocols to implement its security services on a per packet basis, Authenticated Header (AH) and Encapsulated Security Payload (ESP). While the latest version of AH was published without major modifications [16], the current revision of ESP [3] was enhanced with extended Traffic Flow Confidentiality (TFC) and Combined Cipher Modes. The term *IPsec* is used throughout this work to refer to this latest revision of architecture and protocols.

The reader is referred to [17] for an introduction to IPsec or [18] for a review that focuses on cryptographic aspects. For a discussion of the IKE protocol design and alternatives see [19].

5.1 Security Issues

One of the early known security evaluation of IPsec is the comprehensive analysis in [20]. Its authors criticize the complexity of the architecture, point out several design weaknesses and also demonstrate some simple attacks. Their concerns have been confirmed when systematic design flaws where found in IPsec operation modes that use ESP without integrity protection[21, 22]. In addition, concerns about information leakage due to traffic analysis led to the advanced TFC scheme introduced in [23]. This criticism from the academic community found only limited recognition in the IETF, with the result that insecure configurations are still part of the standard and thus also still in operation. Based on these works and our own review of the latest revision of the IPsec standards and implementations, we identify the following security issues in the current specifications.

`/P1/` Encryption-only ESP While previous attacks on ESP encryption without authentication or with AH-based authentication have been blamed on implementations, [22] shows that it is indeed a flaw in the standard to allow encrypted traffic to be unauthenticated or make use of other layers to authenticate the traffic.

`/P2/` Traffic Flow Confidentiality To prevent information leakage via traffic flow analysis, the authors of [23] propose a combination of payload padding, injection of dummy packets and recombination of payloads. But although the problem is acknowledged, only a small subset of the proposed traffic obfuscation mechanisms was standardized in the latest version of ESP. In addition, even these simplified TFC measures are not yet implemented in commodity operating systems like Linux and OpenBSD.

`/P3/` Manual Keying Manual keying poses a serious security threat. It provides no forward or backward secrecy and enables attacks through observation of ciphertext block collisions. IPsec however demands explicit support for manual keying for IPsec packet processing [15], even though any secure key provisioning could be adapted to use the automated keying interface. As a result, popular IPsec instruction guides[4] tend to discuss manual keying in great detail and it must be assumed that such configurations are in widespread use.

`/P4/` Pre-Shared Key (PSK) Authentication The IKEv2 specification [4] describes authentication based on pre-shared keys, a feature that is also much appreciated in IPsec configuration guides[5] and likely in broad use. As also pointed out in the specification, PSK authentication is only secure when keys with high entropy are used. However, although such an assumption does typically not hold when keys are chosen or transmitted by humans, the specification requires ("MUST") PSK support for scenarios where the initiator uses a shared key and the responder uses public-key authentication, clearly a scheme that addresses password-based user authentication. The specification also fails to mention any rate limit behavior to counter brute force attacks on short PSKs.

`/P5/` Pseudo User Authentication The IKEv2 authentication mechanisms themselves are not aware of the type of entity that provides authentication data. The two most

[4] `http://www.ipsec-howto.org/`

[5] `http://wiki.bsdforen.de/howto/ipsec-vpn`

common authentication mechanisms, public-key authentication based on X.509 certificates (PKIX) [24] and pre-shared keys, are both often used to authenticate users. However, none of the major IKEv2 implementations currently support "direct" user authentication. Instead, the shared key or secret key is typically stored on the local disk, in files that are assumed to be accessible only to the platform administrator. Authentication secrets are thus actually used as tokens, allowing simple attacks like theft or use of the access in absence of the user.

`/P6/` Complexity The complexity of IPsec and particularly IKEv1 has often been subject to criticism, for example in [20] and [25]. Subsequent versions of the architecture and particularly IKE include some improvements. However, the complexity of the IPsec architecture and IKEv1 still leads to insecure deployments and even incompatibilities in the key exchange. This complexity is not necessary however. It was noted in [20] already that transport mode is a subset of tunnel mode and that security features provided AH can by large be replaced by a corresponding configuration and application of the ESP protocol. The IKEv2 is much more simple in design than IKEv1, but still has considerable complexity, e.g. in the SA negotiation payloads and the general payload design (wire format). Simpler key negotiation schemes like SKIP [26] and JFK [19] have been proposed, but not accepted for the standard.

`/P7/` Miscellaneous Issues Both versions of IKE employ hash-cookies, simple challenges to efficiently filter spoofed initialization requests, thus mitigating DoS attacks [27]. More advanced protection against DoS is possible by forcing initiators to invest computation resources, but such proposals have been discarded by the IETF due to unclear patent encumbrance[6]. Also, since timeout lengths, retransmission counts and session keep-alive behavior for UDP packet transfer in IKE is not specified, it is trivial to fingerprint IKE servers by observing such behavior [28].

To achieve the security requirements, sVPN has to implement measures against the mentioned problems. Section 6.6 presents measures taken to mitigate `/P1/` to `/P6/`. `/P7/` is not covered in this work however, since any information leaked this way is not considered sensitive. As already mentioned in 3, DoS protection is not necessarily a task of the sVPN service and shall be discussed separately in section 6.2.

6 sVPN Design

The IPsec architecture uses secure channels essentially to bind transferred data to cryptographic identities. This mapping is used to enforce a security policy for each transferred data packet and provides the secure access control and secure channels needed in sVPN. In this section we describes how the sVPN architecture was designed to comply with our requirements and how the IPsec issues discussed in 5.1 are resolved.

6.1 IPsec Virtualization

As described in requirement `/R1/`, sVPN must be able to implement an isolated IPsec boundary for each "protected" area. For a secure IPsec virtualization as defined in re-

[6] `http://www.ietf.org/mail-archive/web/ipsec/current/msg02606.html`

quirements `/S1/` to `/S3/`, sVPN must also be able to identify endpoints of a requested channel and locate the corresponding IPsec policy.

We therefore extend the IPsec policy for sVPN with fields for the source and destination identity of local channels. These identities can be arbitrary labels that are provided to sVPN by other security services. To establish remote channels with other IPsec gateways it must be possible to unambiguously address and authenticate the destination of a channel. In particular, a standard IPsec implementation must be able to request a channel to a specific compartment "protected" by sVPN. To achieve this level of interoperability, existing methods for identification and authentication of peers must be used, which is why sVPN behaves like *multiple* VPN gateways. It uses separate IP addresses for identification on the network layer and separate cryptographic identities for unambiguous authentication during key exchange. If multiple IPs for a single host are not available, NAT (Net Address Translation) can be used to map them to a single address. This limitation is inherent to the IPsec architecture.

6.2 Critical Components

Requirement `/R6/` is achieved by aggressively delegating functionality into the untrusted "protected" and "unprotected" endpoints of each local channel. This process is restricted by `/R7/`, which requires all critical functionality to reside in trusted components, i.e. in the security service in the hypervisor environment. Based on our security requirements, we identify the following functions as critical.

1. The virtualized policy enforcement described in `/S1/` requires that identification, classification and policy matching algorithms are implemented in trusted components.
2. `/S3/` requires that any component that is connected to multiple local channels that should remain isolated must be trusted.
3. To meet requirement `/S5/`, authentication and key exchange via IKEv2 and IPsec traffic processing must be trusted. Availability of key material is reduced by dividing the trusted sVPN security service into two separate modules, the IKEv2 server *iked* and the *ipsec* traffic processing component. As a result, the bulk traffic processing component is isolated from any kind of long-term authentication keys or key negotiation internals.
4. `/S6/` and `/S7/` require the complete IKEv2 key negotiation and IKE SA management to be trusted, as well as the traffic processing components for ESP encapsulation, TFC padding, replay-protection and lifetime management of sessions and keys.

Any remaining functionality is not critical and should be provided by other compartments. Examples are IP routing, packet fragmentation, hardware drivers and NAT traversal. Since the protected and unprotected compartments need to function correctly to establish a connection, DoS protection is only relevant if these are not compromised. Therefore, this feature can be delegated to untrusted components as well.

The rather generic analysis of critical subcomponents is sufficient since our choices in interfaces and components are limited by complexity limitation `/R7/`.

6.3 Usability

We address our usability requirements /R2/ and /R3/ by providing a more abstract VPN service instead of a universal IPsec implementation. This allows us to ignore special use cases of IPsec and make additional assumptions about the user's intentions. As a result, we can ignore the IPsec transport mode as well as AH encapsulation and enforce secure usage of ESP encapsulation. This reduces the complexity of our key negotiation and traffic processing components and facilitates the design of a usable configuration interface. We solve the deployment issue mentioned in /R2/ and /S4/ by leveraging trusted storage, a basic trusted computing service that allows secure transport and storage of information by sealing data to known-good system states. [29]

6.4 Compatibility

As described in /R4/ and /R5/, sVPN is required to provide basic interoperability with other IPsec-based VPNs. We implement this by staying compatible with certain operation modes of IPsec, namely authenticated ESP encryption in tunnel mode, and by providing a IKEv2 implementation that is sufficiently compatible with standard implementations to negotiate this particular configuration. We leverage the flexibility of the IKEv2 negotiation to activate additional non-standard features if supported by the peer. A more detailed discussion of interoperability issues follows in section 8.

6.5 Remote Users

Since VPN access for remote users is one of the main use cases of VPN technology, requirement /R4/ implies support for this scenario in sVPN. Apart from direct user authentication required by /S6/, IP configuration of the remote "protected" compartment is also desirable in this use case. For sVPN, we currently rely on higher level protocols like the Layer 2 Tunneling Protocol (L2TP, see [30]) to provide dynamic IP configuration.[7]

For direct user authentication, we aim to leverage trusted user I/O paths of the operating system to protect the user's login credentials when unsealing the IPsec authentication key from trusted storage. The actual authentication keys are always asymmetric and either imported from an existing PKI or automatically generated by the configuration frontend, which also provides secure deployment. For more complex authentication mechanisms, remote attestation of the peer's TCB will allow to delegate critical functionality to other security services which then vouch for successful authentication of the correct user.

6.6 IPsec Hardening

Section 5.1 discussed several issues in IPsec that need to be resolved to meet our requirements for sVPN. Unfortunately, the manipulation of IPsec internals is limited by

[7] Native IKEv2 support through IKE configuration payloads and traffic selector narrowing is still under investigation. Although simpler in design, these would also add non-critical functionality to the security service.

our compatibility requirements `/R4/` and `/R5/`, with the result that our IKEv2 implementation still suffers from high complexity when compared to our processing component. Nevertheless, our modifications address all identified problems except for fingerprinting and DoS, which we declared minor.

`/M1/` ESP protection To prevent attacks based on `/P1/`, sVPN enforces ESP encapsulation with encrypted authentication for all traffic that is labeled `PROTECT` in the security policy. Weak ciphers are not supported.

`/M2/` Advanced TFC To mitigate `/P2/`, ESP processing supports Traffic Flow Confidentiality (TFC) padding as specified in [4] as well as advanced TFC features described in [23]. Both are negotiated if supported by the peer or enforced by policy.

`/M3/` Manual Keying and PSKs To prevent weak keys and exploitable key management as mentioned in `/P3/` and `/P4/`, our processing component exposes only a low-level interface for automated configuration of keys and policies. To resolve problems with weak PSKs, PSK authentication is not supported in sVPN.

`/M4/` User Authentication As described in 6.5, sVPN authenticates users either through direct user authentication or by leveraging other security services and trusted storage. We expect these mechanisms to supersede the often weak and complex authentication schemes provided by EAP or PSK and to discourage the behavior described in `/P5/`.

`/M5/` Reduced Complexity To mitigate the complexity issues described in `/P6/`, transport mode and AH protection are not supported in sVPN. It implements only the minimal set of functionalities of the IKEv2 specification, which additionally was stripped of authentication in X.509 certificate-based public key infrastructures (PKIX). Instead, sVPN currently authenticates peers based on raw RSA keys and key fingerprint whitelists. Leveraging remote attestation, sVPN may also delegate authentication to trusted remote security services.

7 Implementation

We implemented a prototype as a proof-of-concept, to identify interoperability issues and to estimate complexity of the solution. It encompasses the two critical sVPN components ipsec and iked as well as simple adapters named *tun2ipc* and *udp2ipc* for connectivity with userspace compartments.

Turaya was chosen as the operating system environment. It is open source and implements the PERSEUS security framework based on the L4/Fiasco microkernel and microkernel environment of the DROPS project [31, 32]. With L4Linux, a paravirtualized version of the Linux kernel, L4/Fiasco is also able run Linux systems as guest VMs. Access control for inter-process communication (IPC) and secure identification of compartments through the compartment manager is not yet implemented in Turaya. Our prototype currently simulates this functionality through a local name registration service.

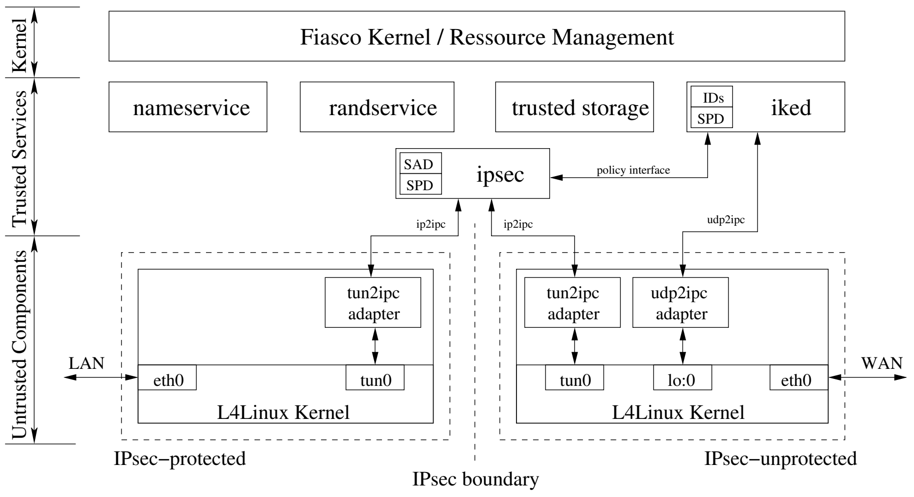

Fig. 2. Detailed architecture of sVPN. A single platform acts as a VPN gateway with two physical network interfaces. The "protected" compartment routes plaintext traffic between the LAN and sVPN. The "unprotected" compartment is responsible for routing the secured data streams into the WAN.

7.1 sVPN Architecture

Figure 2 provides a more detailed view of the sVPN architecture. It shows a platform with two virtualized L4Linux compartments running on top of the Fiasco hypervisor and its environment. The two critical sVPN components reside in the hypervisor environment, next to other security services like trusted storage and the *randservice* random number generator. The two L4Linux compartments have established a local channel through the ipsec module and each also connect to a physical network interface. By configuring the sVPN security policy such that one compartment is in the "protected" and one in the "unprotected" area, the platform is transformed into a VPN gateway, processing traffic between its two interfaces.

Traffic Processing The tun2ipc adapters establish a local point-to-point IP connection between the "protected" and "unprotected" compartments. The connection is set up by connecting to the ipsec module and requesting a forwarding to the destination compartment. The adapters then translate between the socket interface expected by the Linux userspace and the *ip2ipc* IPC interface of the sVPN service. The ipsec module accepts the local connection request only if corresponding policy entries exist in the SPD. In that case, a dedicated *filter thread* is spawned to handles the actual traffic processing for this channel. The filter thread then finalizes the local ip2ipc connection by establishing the second part to the actual destination of the channel. Once the connection is established, each thread is responsible for enforcing the locally relevant subset of the security policy. Interface and implementation are further simplified by making all ip2ipc chan-

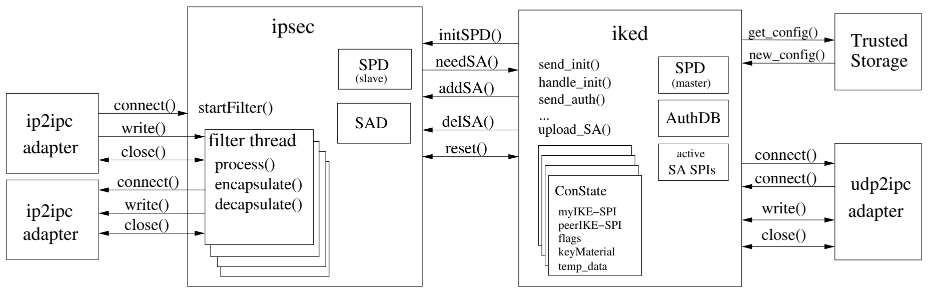

Fig. 3. Schematic overview of the iked and ipsec security modules and IPC channels. Arrows indicate the possible direction of a call, order of calls represents a possible session flow for that interface. The iked component creates a *ConState* structure for each handshake attempt and maintains a list of active SPIs.

nels unidirectional. The destination tun2ipc adapter is responsible for establishing the ip2ipc connection in the other direction, provoking another filter thread to be spawned with corresponding subset of SPD and SAD.

To identify relevant subsets of SPD and SAD, each SPD entry in sVPN also contains two labels identifying source and destination of the ip2ipc channel they apply to, and each SAD entry is linked to an SPD entry. With the relevant parts of SPD, SAD and a unidirectional channel, traffic processing is straight forward. The filter threads have to enforce one of the policy targets (`DISCARD`, `BYPASS`, `PROTECT`) on all traffic they receive, which is trivial to achieve for the first two targets. If `PROTECT` is specified or an encapsulated package is encountered, the packet must be de- or encapsulated with ESP. All required information for ESP processing is contained in the SAD entries available to the thread. If the required SAD entry is not available, the packet is discarded. In case of encapsulation, missing SAD entries additionally trigger a request to the iked module to establish the required SAs. In case of decapsulation, the frames are once more parsed and matched against the SPD to check if the protected environment is allowed to receive this packet. At this point, the SPD would typically specify `BYPASS` or `DISCARD` as the target, but re-encapsulation with a different SA is also possible.

Successfully processed IP frames are forwarded to the adapter of the destination compartment. An ip2ipc adapter or its environment may implement additional uncritical pre- and post-processing of received traffic, like IP (de-)fragmentation, NAT or bandwidth management. Our prototypes simply forward all IP frames between the ip2ipc channel and the L4Linux socket interface.

Key and Protocol Negotiation The key negotiation component iked is a simplified IKEv2 implementation. On startup, it retrieves SPD and long-term authentication keys from trusted storage and establishes the local *policy* and udp2ipc connections to the ipsec and udp2ipc components. The udp2ipc adapter essentially provides a UDP socket to iked, allowing it to send and receive IKEv2 UDP traffic. As can be seen in figure 3, the possible calls for ip2ipc and udp2ipc are very similar, the main difference is that udp2ipc attaches to a UDP socket and that the incoming IPC connection comes from the same iked thread. Like the ip2ipc adapter, it may implement uncritical traffic transformations like defragmentation or even handle IKEv2 protocol features like DoS protection cookies and NAT traversal support. The policy interface between ipsec and iked is used preliminary by iked to initialize the ipsec SPD at startup and manage the ipsec SAD at runtime. The ipsec component only uses it to request a refresh for expired or not yet negotiated SAD entries. Both ends are also able to submit a `reset` command in case database inconsistencies are detected, e.g. when one component was manually restarted.

As IPsec key negotiation does not require high throughput, iked is implemented as a simple single-threaded application. SA negotiation requests from ipsec or remote IKEv2 servers trigger a simple IKEv2 negotiation and that negotiates authenticated encryption with tunnel mode ESP encapsulation and the strongest available cipher suite. Features like NAT detection, DoS protection, endpoint configuration or renegotiation of SAs are not supported. If desired, these should be implement in the untrusted compartments. On success, the resulting SAs pair is uploaded to the ipsec SAD, possibly replacing previous instances for that SPD entry.

7.2 Complexity

We estimate the complexity of the prototype by counting lines of source code (LoC) with SLOCCount[8], which excludes comments from the measurements. As it is custom, our measurements do not include external libraries and cryptographic primitives. However, the two critical components do not make use of use any complex libraries and the cryptographic primitives are comparatively easy to verify. Although the Mikro-SINA project did not document how LoC were measured, their reports for the Viaduct component do not substantially deviate from our own measurement of it with SLOCCount (3,575 LoC). Our prototype has a total code-complexity of 5,187 LoC including adapter components. The critical subcomponents iked and ipsec have a complexity of only 2,919 and 839 LoC, respectively, plus 917 LoC for common SPD and SAD management. Although our prototype still misses some important features, the difference to standard IPsec implementations is impressive. The Mikro-SINA adaption of the isakmpd IKEv1 server in [12] has an estimated 22,800 critical LoC if unnecessary components were to be removed. We counted 46,600 and 30,800 LoC for the IKEv1 and IKEv2 implementations of the strongSwan project, plus 30,000 lines of IKE specific libraries. Our ipsec module is significantly less complex than the Mikro-SINA Viaduct (839+917 vs. 3,575 LoC), since we exclude transport mode and AH encapsulation. For comparison, a sample of obviously IPsec related files in Linux 2.6.26[9] results in a similar figure of about 3,500 LoC.

7.3 Current Status

Our prototype successfully establishes ESP tunnels with a standard IPsec implementation and the strongSwan IKEv2 server. The connection was established with PSK authentication however, since raw public key authentication is not yet implemented in iked

[8] SLOCCount by David A. Wheeler: `http://www.dwheeler.com/sloccount/`

[9] include/net/{ip.h,ah.h,esp.h,xfrm.h,ipcomp.h} net/ipv4/{esp4.c,ah4.c,xfrm4*,ipcomp.c}

and strongSwan. The negotiated SAs for IKEv2 and ESP use AES-128, SHA-1 and the standard Diffie-Hellman groups from [4]. The current implementation still misses some necessary functionality like timeout handling and public key authentication for iked and advanced TFC for ipsec. Still, we do not expect their complexity to rise anywhere near the size of standard implementations.

8 Interoperability Issues

The IPsec modifications `/M1/` to `/M5/` and the untypical compartmentalized architecture potentially result in interoperability problems with standard IPsec implementations. As will be seen in this section however, most modifications are simply a matter of appropriate configuration of the IPsec peer. We shall also discuss some optional features like IPComp and NAT detection that conflict with our compartmentalized design approach.

IPsec Modifications To reduce implementation complexity and improve security, section 6.6 specified modifications to the IPsec implementation in sVPN. As described below however, most of these modifications are only a question of correct configuration so that interoperation with standard IPsec implementations is still possible.

`/M1/` Authenticated Encryption and Primitives Employed cryptographic algorithms and their combination are by design negotiated during key exchange and subject to SPD configuration. Standard IPsec implementations tend to include as many strong and required suites as possible, increasing the chance to find a common cipher suite.

`/M2/` Advanced TFC Padding Similarly to the extended TFC, support for advanced TFC padding can be negotiated during IKEv2 key exchange. It thus does not interfere with standard implementations except when enforced by sVPN policy.

`/M3/` Manual Keying and PSK Manual key provisioning and pre-shared key authentication are alternative key distribution models supported by IPsec, along with public key authentication and others. sVPN in contrast supports only public key authentication. However, any resulting interoperability problems are an intended trade off for complexity and security. If desired, it is possible to use alternative key management systems instead and directly interface with the policy IPC interface of the ipsec module.

`/M4/` No direct User Authentication Although identity types are supported in the IKEv2 protocol, the authentication mechanism is ignorant of the type of entity that is authenticated. However, direct user authentication can obviously not be enforced on IPsec implementations that do not support remote attestation or direct user authentication. The enhanced user authentication can only be beneficial to the platform that supports it, possibly decreasing the security of the overall network security.

`/M5/` Transport Mode, AH, raw RSA Keys Missing support for transport mode and the AH protocol should not result in unsolvable interoperability issues since their use is a matter of policy configuration and negotiated via IKE. The same applies for raw RSA key authentication, although this authentication mode is less commonly supported by other IKEv2 implementations.

Automated NAT Traversal IPsec uses UDP encapsulation of ESP payloads for compatibility with NAT, adding 8 bytes of overhead per packet to traverse NAT routers[33, 34]. In IKEv2, existence of NAT is automatically detected during SA negotiation and UDP encapsulation is activated. In sVPN however, udp2ipc abstracts from the UDP socket interface and currently just provides a channel ID instead of ports and addresses, relieving iked from any layer 3 protocol handling. Additionally, NAT detection is not a critical feature and should ideally be implemented in untrusted components.

From a security perspective, the best solution may be to add fake NAT detection payloads into the IKEv2 exchange and thus transform the automated detection into a manual configuration option controlled by the adapters. Alternatively, the udp2ipc adapters could provide network layer information to iked when establishing a new udp2ipc channel, thus enabling iked to implement NAT detection.

IP Fragmentation As discussed in section 7 of the specification in [15], the IPsec architecture has systematic issues with fragmentation. For sVPN, the relevant places where fragmentation may be encountered are ESP encapsulation and decapsulation, where header information may be missing for correct policy matching of packets or authentication of ESP fragments may fail. Additionally, the ip2ipc channel also imposes a limitation to the size of packets that may have been defragmented to larger frames in the untrusted compartment.

Our approach is similar to the suggestions in [15]. At startup, our ipsec processing thread searches their sub-SPD for any entries that require knowledge of higher layer protocols. If such entries are not encountered, any kind of packet can be parsed according to policy. Otherwise, it will drop any fragmented packets and leave defragmentation to the sending adapter or its environment. For successful decapsulation, the security service has no choice but to rely on the sending compartment to defragment the ESP frames. Fragmented frames are dropped based on IP header information, since they will always fail to authenticate. The behavior is optimized by setting the `Don't Fragment` bit in the IP header of outgoing ESP frames. The Maximum Transmission Unit (MTU) of the ip2ipc IPC channel is also handled in the adapter components. They must watch the size of defragmented ESP frames or other payloads and inform the sender on the limited maximum segment size if required.

IP Compression Traffic compression is usually applied transparently at the link layer, but data encryption renders compression algorithms ineffective. [35] therefore specifies an IP compression standard IPComp to supplement IPsec. IPComp compresses IP payloads, inserting itself as a logical protocol layer between the payload and the IP layer before encryption of the packet. As a non-critical transformation of traffic, it should be implemented in the tun2ipc adapters. However, IPComp changes the information available to the packet parser. The protocol type is changed to indicate a compressed payload and no transport layer information is available without decompression. Therefore, IPComp has to be applied between processing steps in the ipsec component and can not be delegated to untrusted components.

9 Security Considerations

Since the ip2ipc and udp2ipc interfaces are semantically similar to those of standard IPsec implementations and since sVPN essentially implements and enforces the use of a subset of alternative IPsec mechanisms, one might argue that the security of sVPN can be reduced to at least that of standard IPsec implementations. In the following we show informally why sVPN additionally fulfills the stronger requirements described in 3.2.

`/S1/` For local IPC channels, identification of the endpoints is an implementation detail that should be solved by the operating system's IPC mechanisms. We also extended the standard SPD entries by two fields to specify source and destination identity each rule applies to. Even without OS support, this already allows us to implement secure identification by providing random unique identification labels to the untrusted compartments at startup time. Requirement `/S1/` is thus fulfilled.

`/S2/` We fulfill this requirement since we assume that the hypervisor enforces isolation between all local compartments and channels. It follows directly that any interconnection of compartments is thus be routed through sVPN or other trusted services.

`/S3/` This requirement is fulfilled by a correct implementation of the trusted service, i.e. logically isolated processing of channels in the service. We implemented this by launching a separate processing thread for each local channel. The threads do not share any writeable resources; they could even be encapsulated in separate L4 tasks so that the memory isolation is enforced by the kernel.

`/S4/` This requirement is fulfilled since sVPN is designed to retrieve its IPsec policy and authentication secrets directly from trusted storage. The sealing functionality of trusted computing systems protects the configuration data using public key cryptography and local attestation of the environment that requests accesses to the data.

`/S5/` As discussed in section 6.2, we implement all critical functionality, including all functionality that requires long-term authentication secrets or session keys, in the sVPN security service. Keys are only transmitted inside this module or in a local isolated channel between sVPN and trusted storage. The interfaces exposed by sVPN are equivalent to those exposed by standard IPsec implementations. Based on the security of the IKEv2 and ESP protocols, it follows that no covert channels exist to retrieve key material. Additionally, we reduced the availability of short- and long-term keys to the required subcomponents.

`/S6/` As a trusted component in the security services layer, our service is able to directly connect to other trusted security services. These in turn are able to establish physical presence of a user or to enforce arbitrary authentication schemes. Although not yet implemented in our prototype, our design thus fulfills `/S6/`.

`/S7/` We hardened our traffic processing based on our review of IPsec security issues in section 5.1. Additionally, we aim to provide optional advanced protection against traffic flow analysis (advanced TFC). Except for possibly weak authentication modes, which we addressed through direct user authentication in section 6.5, there are no relevant security issues with IKEv2. There are thus no known problems with the secure channels provided by sVPN.

10 Conclusion

We proposed an adaption of the IPsec architecture for VPNs in trusted computing environments. In accordance with the PERSEUS security concepts, we isolated critical functionality into self-contained subcomponents of minimal complexity. We solved several security issues of IPsec by reducing the framework to a simple VPN service and discussed how additional features can be integrated in a compatible fashion.

The result is a high-security VPN service that should provide a high usability. The architecture solves the conflict of interest between owner and user in the remote user use case, allowing companies to deploy machines that are highly secure regarding access to the company VPN and applications, but use flexible compartments to suite the needs of the employees. Its small code-size and modular design make sVPN an ideal target for future research on secure channels in trusted environments.

11 Further Work

As noted in section 7.3, our implementation is far from complete. We also postponed investigation of IKE mobility extensions [36] , resistance against timing-based side-channel attacks and detailed performance optimizations.

Remote Attestation The Trusted Network Group (TNG) proposes an extensive framework for remote attestation in [37, 38]. However, these specifications seem to add significant overhead to sVPN and we question if this level of flexibility is necessary in the first place.

Roaming Users In context of the remote user use case `/U2/`, the MOBIKE [36] extensions to IKE and its adaption to the compartmentalized sVPN architecture should be investigated. Use of TS narrowing and configurations payloads combined with a good user authentication scheme may eliminate the need for additional higher layer protocols.

Simplified PKI The several PKIX-related standards published by the IETF document the complexity of this authentication scheme. We currently favor raw RSA keys instead, but these miss a key hierarchy to support large environments. It is unclear if and how a compromise can be compatible with standard implementations.

References

1. Sadeghi, A.R., Stüble, C.: Bridging the Gap between TCPA/Palladium and Personal Security. `citeseer.ist.psu.edu/575430.html` (2003)
2. Helmuth, C., Warg, A., Feske, N.: Mikro-SINA - Hands-on Experiences with the Nizza Security Architecture. In: Proceedings of the D.A.CH Security. (March 2005)
3. Kent, S.: IP Encapsulating Security Payload (ESP). RFC 4303, Internet Engineering Task Force (December 2005)
4. Kaufman, C.: Internet Key Exchange (IKEv2) Protocol. RFC 4306, Internet Engineering Task Force (December 2005)

5. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: CRYPTO 1996. (1996) 104–113
6. Percival, C.: Cache missing for fun and profit. In: Proceedings of BSDCan 2005. (2005)
7. Sailer, R., Valdez, E., Jaeger, T., Perez, R., van Doorn, L., Griffin, J.L., Berger, S.: sHype: Secure Hypervisor Approach to Trusted Virtualized Systems. Research Report RC23511, IBM Research (February 2005)
8. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: A Virtual Machine-Based Platform for Trusted Computing. In: Proceedings of the 9th ACM Synopsium on Operating System Principles. (2003) 193–206
9. Shapiro, J., Hardy, N.: EROS: A Principle-Driven Operating System from the Ground Up. IEEE Software (January 2002) 26–33
10. Härtig, H., Hohmuth, M., Feske, N., Helmuth, C., Lackorzynski, A., Mehnert, F., Peter, M.: The Nizza Secure-System Architecture. In: Proceedings of IEEE CollaborateCom 2005, IEEE Press (December 2005) 10pp.
11. Heiser, G., Elphinstone, K., Kuz, I., Klein, G., Petters, S.M.: Towards Trustworthy Computing Systems: Taking Microkernels to the Next Level. In: ACM Operating Systems Review, 41(4). (July 2007) 3—11
12. Syckor, J.: IPSec Infrastruktur für Mikro-SINA. `os.inf.tu-dresden.de/papers_ps/syckor-diplom.pdf` (November 2004) Diplomarbeit, Technische Universität Dresden.
13. McDonald, D., Metz, C., Phan, B.: PF_KEY Key Management API, Version 2. RFC 2367, Internet Engineering Task Force (July 1998)
14. Atkinson, R.: Security Architecture for the Internet Protocol. RFC 1825, Internet Engineering Task Force (August 1995)
15. Kent, S., Seo, K.: Security Architecture for the Internet Protocol. RFC 4301, Internet Engineering Task Force (December 2005)
16. Kent, S.: IP Authentication Header. RFC 4302, Internet Engineering Task Force (December 2005)
17. Doraswamy, N., Harkins, D.: IPsec: The new Security Standard for the Internet, Intranets and Virtual Private Networks (second edition). Prentice Hall PTR (2003)
18. Paterson, K.G.: A Cryptographic Tour of the IPsec Standards. `citeseer.ist.psu.edu/737404.html`
19. Aiello, W., Bellovin, S.M., Blaze, M., Ioannidis, J., Reingold, O., Canetti, R., Keromytis, A.D.: Efficient, DoS-resistant, secure key exchange for internet protocols. In: CCS '02: Proceedings of the 9th ACM conference on Computer and communications security, ACM (2002) 48–58
20. Ferguson, N., Schneier, B.: A Cryptographic Evaluation of IPsec. `www.schneier.com/paper-ipsec.html` (2000)
21. Bellovin, S.: Problem Areas for the IP Security Protocols. In: Proceedings of the Sixth Usenix Security Symposium. (July 1996)
22. Degabriele, J.P., Paterson, K.G.: Attacking the IPsec Standards in Encryption-only Configurations. eprint.iacr.org/2007/125 (2007)
23. Kiraly, C., Bianchi, G., Formisano, F., Teofili, S., Lo Cigno, R.: Traffic masking in IPsec: architecture and implementation. In: Mobile and Wireless Communications Summit, 16th IST. (July 2007) 1–5
24. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, Internet Engineering Task Force (May 2008)
25. Simpson, W.A.: IKE/ISAKMP Considered Harmful. Usenix ;login, Volume 24, Number 6 (December 1999)
26. Aziz, A., Patterson, M.: Design and Implementation of SKIP. In: INET'95 Hypermedia Proceedings. (1995)

27. Kaufman, C., Perlman, R., Sommerfeld, B.: DoS protection for UDP-based protocols. In: CCS '03: Proceedings of the 10th ACM conference on Computer and communications security, ACM (2003) 2–7
28. Izadinia, V.D., Kourie, D., Eloff, J.: Uncovering identities: A study into VPN tunnel fingerprinting. In: Computers & Security, vol.25, issue 2. (2006) 97–105
29. Group, S.W.: Tcg storage architecture core specification. Technical report, Trusted Computing Group (May 2007)
30. Lau, J., Townsley, M., Goyret, I.: Layer Two Tunneling Protocol - Version 3 (L2TPv3). RFC 3931, Internet Engineering Task Force (March 2005)
31. Alkassar, A., Stüble, C.: Die Sicherheitsplattform Turaya. In: Trusted Computing, Vieweg+Teubner (May 2008) 86–96
32. Härtig, H., Roitzsch, M.: Ten years of research on l4-based real-time systems. In: Proceedings of the Eighth Real-Time Linux Workshop. (2006)
33. Aboba, B., Dixon, W.: IPsec-Network Address Translation (NAT) Compatibility Requirements. RFC 3715, Internet Engineering Task Force (March 2004)
34. Huttunen, A., Swander, B., Volpe, V., DiBurro, L., Stenberg, M.: UDP Encapsulation of IPsec ESP Packets. RFC 3948, Internet Engineering Task Force (January 2005)
35. Shacham, A., Monsour, R., Pereira, R., Thomas, M.: IP Payload Compression Protocol (IPComp). RFC 2393, Internet Engineering Task Force (December 1998)
36. Eronen, P.: IKEv2 Mobility and Multihoming Protocol (MOBIKE). RFC 4555, Internet Engineering Task Force (June 2006)
37. Group, T.W.: Architecture for interoperability. Specification, Trusted Computing Group (April 2008)
38. Group, T.W.: If-t: Protocol bindings for tunneled eap methods. Specification, Trusted Computing Group (May 2007)